Ad Hoc Human Information Nets for Asymmetric Threat Surveillance

18 June 2003

Sponsored by

Defense Advanced Research Projects Agency (DOD)

ARPA Order K475/73

Issued by U.S. Army Aviation and Missile Command Under
Contract: DAAH01-03-C-R048
Final Report
Reporting Period:
5 February 2003 — 18 June 2003

Technical Report NIM-2003-002

Prepared by:
George K. Thiruvathukal, Ph.D.
Nimkathana Corporation
1807 W. Winnemac Avenue, Unit A
Chicago, IL 60640
312.515.2560
gkt@acm.org

1 Overall Status

This is the final report for the Ad Hoc Human Interaction Nets project. All tasks described in our original proposal have been completed: Hybrid-Hierarchical Micro-Databases, Query Optimization, and XML for Embedded Systems. After meeting with the program manager, Tom Armour, in Washington, DC, we were able to get a better perspective on the ad hoc nature of the solicitation. Armed with this perspective, we were able to complete two substantial prototypes to address the four tasks as originally written that works with Pocket PC and Palm OS devices. We were able to evaluate the viability of the Sharp Zaurus platform as well but ultimately decided to limit our focus to Palm OS and Pocket PC devices, which appear to have the greatest market interest and viability at the time of writing. Acquiring the hardware enabled us to make substantial progress on the *ad hoc* networking issues. In our networking prototype, which is based on the Pocket PC platform, we were able to take advantage of the emerging .Net compact framework and use it to build a distributed services and database framework. In this report, we will describe the two concept prototypes that were developed to demonstrate the feasibility of ad hoc human interaction networks and then proceed with a brief discussion of each task as written in the original proposal.

2 Prototypes Completed

2.1 Ad Hoc XML Capture and Editing on Handheld Devices

It is well established in both research and industry that XML will play a key role in data processing in the future. What is not so well established is whether it will ever become usable in the same way that a word processor is usable, especially on handheld devices. Authoring XML documents requires the use of tools that, at best, present a cumbersome experience for non-technical users. In particular, such users must be familiar with concepts they would rather know little or nothing about, such as markup syntax, schema languages (which come in many varieties), parsers, and validation. Then there is the matter of being able to do XML processing on handheld devices themselves. XML, which emphasizes the creation and manipulation of document trees, does not lend itself to efficient use on any handheld platform on the market today.

We completed the development of a tool that can take an XML schema, encoded in XML itself, and generate code for forms that can be used to capture and edit XML documents on handheld devices. The tool presents a number of innovations that, for all practical purposes, results in XML never actually being used in the process. Documents are maintained in a parsed form, similar to what is found in compilers, which allows new or existing documents to be created or modified, respectively, without having to keep the entire document in memory. Given that today's handheld devices have limited on-device memory but can be expanded with flash memory (e.g. Secure Digital or CompactFlash cards), the approach makes sense. Documents can be stored in our out of the device's physical memory, and documents can be edited simply by bring document fragments into and out of memory on demand.

As an example, consider the small fragment of XML, which is a simplified schema dialect we created explicitly for the purpose of ad hoc data definition. It is fully equivalent to the native XML DTD concept but easy to parse using a standard XML parser:

```
<DocumentType>
   <sequence>
   <element name="documentation" root="yes">
     <sequence>
       <element name="incident" cardinality="1">
         <sequence>
           <element name="location" cardinality="1">
             <attribute name="building" use="#IMPLIED">
                <value value="Damen Hall"/>
                <value value="Mertz"/>
                <value value="Campion"/>
                <value value="West A"/>
             </attribute>
```

```
                <attribute name="campus" use="#IMPLIED">
                   <value value="Lakeshore"/>
                   <value value="Watertower"/>
                   <value value="Rome"/>
                </attribute>
                <pcdata/>
              </element>
<!- This schema has been significantly abbreviated for the purpose of
presentation.  ->
             <element name="action" cardinality="*">
                <pcdata/>
             </element>

              <element name="comment" cardinality="*">
                <pcdata/>
              </element>
             </sequence>
           </element>
         </sequence>
       </element>
       </sequence>
    </DocumentType>
```

It is beyond the scope of a progress report to introduce every detail of the schema shown above; however, the schema defined here is inspired by the notion of an incident report, which is used by first responders in day-to-day inicidents that occur normally. Here we are defining an incident report structure that may be used for on-campus security at a university. In a nutshell, this schema defines the following:

1. The top level, DocumentType, is used to identify the encoding language for this schema. We have developed a tool that actually generates this schema from XML's intrinsic schema known as a Document Type Definition (DTD).

2. <sequence> is used to define a sequence of elements that may appear in succession.

3. <element> is used to define an element. The very first type of element expected is one named documentation. As will be shown, this results in a form being created to capture documentation.

4. <attribute> is used to define attributes associated with a particular element. Attributes in our system can either be defined with arbitrary character data (the <cdata/> descriptor) or an enumerated list of values <value/>. When attributes are defined with values, these are translated into drop-down lists on forms. When attributes are defined to contain arbitrary character data, they are translated into a simple text entry field.

5. <pcdata> allows character data that would appear within an <element> definition to be defined. This results in a multi-line text field being rendered on the form without requiring an attribute to be defined.

In short, the schema language presented here is fully equivalent to the native DTD found within XML, but has been augmented with a number of features to facilitate rendering forms that are found, typically, on handheld operating systems (notably the Palm OS). Figure 1 shows screen captures for a few of the forms generated for the C version of our tool for the incident-capturing schema shown above. (Other forms are shown subsequently for the Java version.)

There are many forms generated, and the output forms are compatible with the standard Palm tools, most notably the PiLRC tool, which is the resource definition language for creating forms on Palm OS. The item most noteworthy of mention is that changing the schema allows a completely new interface and capture/edit application to be generated without writing a single line of code.

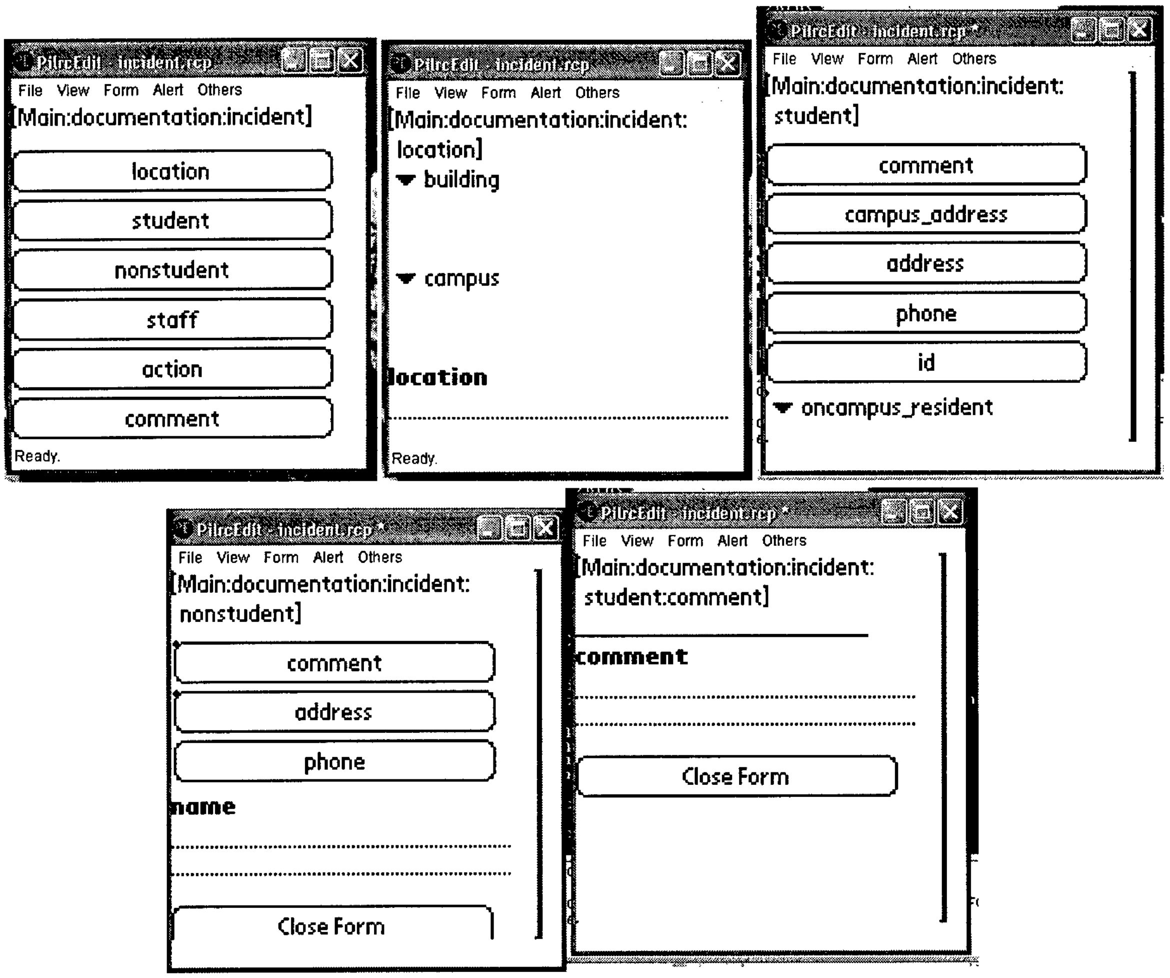

PilrcEdit - incident.rcp
File View Form Alert Others
[Main:documentation:incident]
location
student
nonstudent
staff
action
comment
Ready.
PilrcEdit - incident.rcp
File View Form Alert Others
[Main:documentation:incident: location]
building
campus
location
Ready.
PilrcEdit - incident.rcp *
File View Form Alert Others
[Main:documentation:incident: student]
comment
campus_address
address
phone
id
oncampus_resident
PilrcEdit - incident.rcp *
File View Form Alert Others
[Main:documentation:incident: nonstudent]
comment
address
phone
name
Close Form
PilrcEdit - incident.rcp *
File View Form Alert Others
[Main:documentation:incident: student:comment]
comment
Close Form

Figure 1: Screenshots from Schema to Form Generation Tool

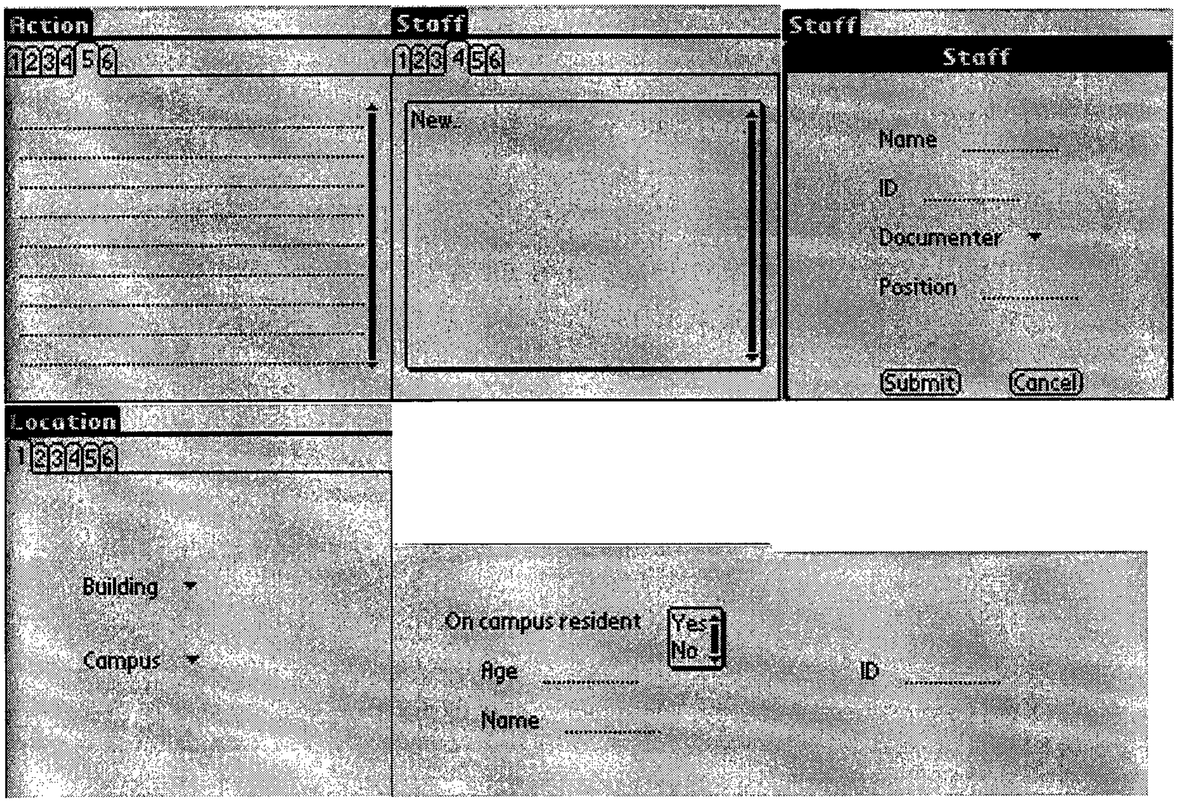


Figure 2: Screen Shots from the SuperWaba Java Version

In addition to generating forms, the tool we have developed and are continuing to evolve can directly save and edit documents that are embedded in Palm databases. The advantage of the approach is that large documents can be captured without using all of the available system memory. When a fragment of XML is being captured with a form, the fragment is loaded into memory, the form is populated, and saved when the user is finished. We are also exploring optimizations to allow only one form to be active at a time by keeping track of the XPath (XML's framework that allows one to know where he/she is when considering a particular document fragment).

The translation tool was developed using the Python language, and the code generated is a combination of PiLRC and ANSI C code.

We have also been exploring the translation into Java, specifically the SuperWaba Java development kit for Palm, which features a robust library for programming Palm and PocketPC applications. We have not yet integrated this into our translation tool but have figured out the technical issues required to do so by performing a hand-translation. Figure 2 illustrates screen shots from a sample application that is actually running on the Palm devices with the SuperWaba Java Virtual Machine (JVM) installed.

One advantage we have observed in moving to SuperWaba is the ability to take advantage of a number of higher-level widgets. In particular, tabbed forms are quite useful for working with XML, where there is often the need to visit multiple children from a given "element" definition. As well, working with Java in general results in greater reliability by supporting automatic memory management, a clean event model, and in general easier development, debugging, and testing. We were only able to spend the last month with the SuperWaba platform but intend to continue working with it as part of future work.

2.2 Distributed Data Capture, Consolidation, and Query Processing on Handheld Devices

Exploration of .Net The Microsoft .Net Compact Framework (hereafter, .Net Compact) is a brand new initiative to provide great customer satisfaction any time, any place and on any device. We were only able to spend two months evaluating its feasibility, since it was released in late-April 2003.

The .Net compact framework brings XML web services to handheld/wireless devices The .Net compact framework is a subset of full desktop .Net framework, yet it provides most of the functionality required for ad hoc networking. We have limited the use of .Net on the Pocket PC to client-side only. The .Net Compact framework does not yet support services that are hosted on the devices themselves. From the standpoint of application development, the programming entails many common aspects with ordinary client/server programming.

The .Net Compact provides a broad range of superb features that one can take advantage in building networking applications. It provides two namespaces, System.Net and System.Net.Sockets for low-level networking programming. The System.Net classes are similar to Microsoft's WinSock or WinInet APIs (used for sockets programming), in that they allow applications to get and send data using the TCP and UDP (datagram) Internet protocols. We explored these low-level aspects but ultimately decided to go with the higher-level XML RPC (remote procedure call) approach called SOAP (simple object activation protocol), which allows us to take advantage of remote database calls through another component framework known as ADO.Net.

Recalling from our first progress report, where we presented details of the data capturing application, the addition we made here was to consolidate the data on the PocketPC PDA. Using the .Net Compact framework and SOAP, we were able to migrate the consolidated data to the server. The idea of how part of this works is shown in figure 3.

In this figure, a PDA can publish its data to a remote server, simply by constructing a SOAP request message, which is yet another area of this project where we take advantage of XML (an entirely different use of XML). Looking at the request message (shown toward the bottom of the figure), a remote service, which we have developed, accepts at any time new published measurements (or other ad hoc content) from a PDA. These requests can be done using the HTTP protocol itself and allows the client to maintain a fairly lightweight protocol stack for communication.

In addition to a *publication* service, we were able to build other services for registration, discovery, and distributed queries. The elegance of network services in the context of this application cannot be overstated, and .Net Compact (for a nascent technology) appears to provide a feasible platform for exploring distributed services and data processing. We hope in future work to evaluate the scalability, which has not yet been determined and was beyond the scope of our proposed effort.

Finally, we did consider the use of Jini from Sun Microsystems. Unfortunately, Jini as a technology is undergoing a major rethinking at Sun. Given its strong dependence on Remote Method Invocation (RMI) and running multiple JVM instances on the devices as well as the servers, it is not a viable solution. In our experiments, Jini requires a fairly robust server with significant RAM, even to run the most straightforward Jini services. We were able to explore the proposed ideas, nevertheless, simply by considering Microsoft .Net Compact. As part of our future plans, we intend to explore other XML RPC frameworks, such as XML/RPC from `http://www.xmlrpc.org`, and will allow for greater interoperability among all of the devices under consideration. Microsoft .Net Compact limits one to the Microsoft platform.

3 Milestone/Task Status

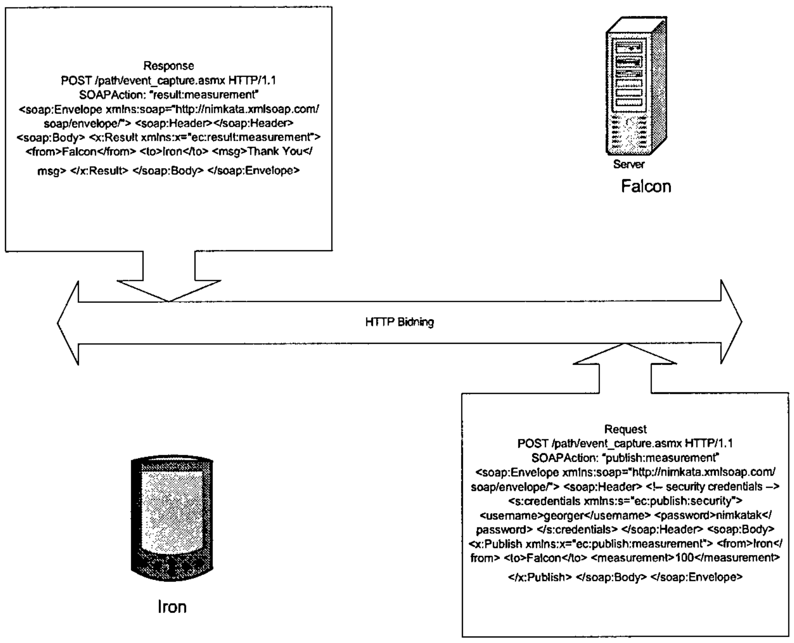
3.1 Overview

The remaining subsections in this section describe progress on the 4 major tasks. For each task, we describe the following:

Schedule Whether the task is on schedule. We use the term **in progress** to describe a task that is partially or nearly completed and **not started** to describe a task that has not been started

% Completion An estimate of the percentage completed for any task **in progress**.

Testing Program Not applicable to this project.



```
Response
POST /path/event_capture.asmx HTTP/1.1
SOAPAction: "result:measurement"
<soap:Envelope xmlns:soap="http://nimkata.xmlsoap.com/
soap/envelope/"> <soap:Header></soap:Header>
<soap:Body> <x:Result xmlns:x="ec:result:measurement">
<from>Falcon</from> <to>Iron</to> <msg>Thank You</
msg> </x:Result> </soap:Body> </soap:Envelope>
```

```
Request
POST /path/event_capture.asmx HTTP/1.1
SOAPAction: "publish:measurement"
<soap:Envelope xmlns:soap="http://nimkata.xmlsoap.com/
soap/envelope/"> <soap:Header> <!-- security credentials -->
<s:credentials xmlns:s="ec:publish:security">
<username>georger</username> <password>nimkatak</
password> </s:credentials> </soap:Header> <soap:Body>
<x:Publish xmlns:x="ec:publish:measurement"> <from>Iron</
from> <to>Falcon</to> <measurement>100</measurement>
</x:Publish> </soap:Body> </soap:Envelope>
```

Figure 3: Basic Approach Used to Access/Serve a .Net Compact Framework Service using SOAP

Designs Completed Summary of designs and implementations. In the context of software-related projects, we interpret designs as prototypes developed thus far and any supporting design/architectural sketches.

3.2 Hybrid-Hierarchical Database Model

Schedule

Completed.

% Completion

100%+

Testing Program

Not applicable.

Designs Completed

We were able to complete the data consolidation aspects of this task by using the .Net Compact Framework on the Pocket PC platform.

3.3 Query Processing

Schedule

Completed.

% Completion

100%

Testing Program

Not applicable.

Designs Completed

We were able to complete this task fairly easily. As it turned out, the work we performed in support of the next task, provided much of the infrastructure we needed for distributed query processing. We focused our efforts, given this was to be a feasibility study, on the Pocket PC platform, which provides support for remote query processing (on servers or other devices) through the .Net Compact framework.

In future work, we hope to use XML/RPC on the Palm devices to develop a similar prototype based on the Linux (or any Unix) server platform and open-source databases such as MySQL or PostgreSQL.

3.4 Distributed Services

Schedule

Completed.

% Completion

100%

Discussion

We developed a distributed services framework using the .Net Compact framework to address issues of discovery, registration, data consolidation, and query processing. The prototype discussed in section 2.2 was completed in direct support of this task. We also explored XML/RPC programming on the Palm platform and intend to develop a mirror of our functional prototype as part of future effort.

3.5 XML Processing

Schedule

Completed.

% Completion

100%

Designs Completed

As described in the first progress report, the work remaining to be done was to integrate the XML embedding framework with the form generation application. We were able to use this framework on the devices as the storage format for creating/editing XML documents.

4 Outstanding Issues from Previous Report

None. We completed all of the tasks and were able to demonstrate substantial progress to our program manager during a visit to Washington, DC.

5 New Problems

None.

6 Conferences and Trips

The PI (Thiruvathukal, President/CEO) and senior personnel (Alok N. Choudhary, Chief Scientist and Director) travelled to Washington, DC, on 30 April 2003 to present our work to Tom Armour, who was the program manager associated with the solicitation topic.

7 Any Potential Impacts on Schedule

None.

8 Conclusion and Future Plans

We have developed two significant prototypes to demonstrate the feasibility of ad hoc human interaction networks. We explored two interpretations of the notion of *ad hoc*. The first interpretation is ad hoc data definition and capture. The second is ad hoc networking, data consolidation, and distributed query processing with network services. We believe these two prototypes can be evolved into software that is relevant from both a commercial and an academic research perspective.

We remain hopeful to pursue a Phase II SBIR or other contact within the Department of Defense and DARPA. There appear to be a number of solicitations and needs where the work presented herein could have a natural evolution path, including schemaless data processing, the LifeLog project, and MyDay, just to name a few. As well, the Palm and PocketPC communities are both struggling to make XML usable on their devices and need technologies and techniques that will allow XML to be used efficiently and intuitively. This SBIR contract has given us an opportunity to develop significant in-house expertise in handheld/wireless development, which is one of the few software areas in high demand in the challenging technology environment/economy being faced in the U.S.A. and worldwide.

9 Itemized Person-Hours and Costs

During this progress period, the PI, senior personnel, and student interns completed the prototypes and tasks described.

Thiruvathukal Principal Investigator and Lead Architect 190 hours at $100/hour ($19,000)

Alok Choudhary 40 hours at $100/hour ($4,000).

Student Interns Peter Nabicht, Karim Kabani, and Chris Stump worked a total of 855 hours at $20/hour. ($17100)

The total labor cost for this progress period was $40,100.

9.1 Equipment

The following equipment purchases were made to support the unique development requirements of this project. For each item, the quantity, unit cost, and subtotal are listed in parentheses. No justification is provided, since these items are consistent with the budget and justification presented in the original proposal.

- 2 Viking 256MB CF Cards (2, $55, $101.00)
- 2 Sharp SL-5500, (2, $348, $696.00)
- Linksys Wireless CF Card (2, $62.50, $125.00)
- EDGE 256MB SD Card (4, $76.00, $304.00)
- Tungsten T (2, $343.00, $686.00)
- Toshiba e750 (2, $560.00, $1,120.00)
- Samsung Syncmaster Server Monitor (1, $605.00, $605.00)
- LAN Switch (2, $194.00, $388.00)
- Wireless LAN (2, $115.30, $230.60)
- Bluetooth Adapters (4, $76.50, $306.00)
- Compaq Developer Laptop (2. $1,600, $3,200.00)
- Apple Developer Laptop (1, $2,540, $2,539.86)
- Linux Server (1, $2,800, $2,800.00)
- Windows Server for .Net (1, $500, $499.98)
- KVM Switch (1, $198, $198.00)
- Mice (3, $39, $117.00)
- Palm Tungsten C (2, $435, $870.00)
- Palm Zire (2, $265, $530.00)

The total equipment costs for this progress period was $15,316.44.

MATERIAL INSPECTION AND RECEIVING REPORT

Form Approved OMB No. 0704-0248

PLEASE DO NOT RETURN YOUR COMPLETED FORM TO THE ABOVE ADDRESS.
SEND THIS FORM IN ACCORDANCE WITH THE INSTRUCTIONS CONTAINED IN THE DFARS, APPENDIX F-401.

1. PROCUREMENT INSTRUMENT IDENTIFICATION (CONTRACT) NO. DAAH01-03-C-R048	ORDER NO. N/A	6. INVOICE NO./DATE N/A	7. PAGE 1 OF 1	8. ACCEPTANCE POINT D
2. SHIPMENT NO. NIM0002X	3. DATE SHIPPED 2003JUN21	4. B/L N/A TCN N/A	5. DISCOUNT TERMS N/A	

9. PRIME CONTRACTOR CODE 3B2H9 Nimkathana Corporation 1807 W. Winnemac, Unit A Chicago, IL 60640	10. ADMINISTERED BY CODE S1403A DCMA Chicago 1523 W. Central Road Arlington Heights, IL 60005
11. SHIPPED FROM *(If other than 9)* CODE FOB:	12. PAYMENT WILL BE MADE BY CODE HO0339 DFAS - Columbus Center/HQ0339 West Entitlement Operations P.O. Box 182381 Columbus, OH 43218-2381
13. SHIPPED TO CODE W31P4O Earnest Taylor, Jr./W31P4Q Redstone Arsenal, AL 35898-5280 earnest.taylor@redstone.army.mil	14. MARKED FOR CODE N/A

15. ITEM NO.	16. STOCK/PART NO. DESCRIPTION *(Indicate number of shipping containers - type of container - container number.)*	17. QUANTITY SHIP/REC'D*	18. UNIT	19. UNIT PRICE	20. AMOUNT
0001 AB	Final Report	1	LO	45,406.00	45.406.00

21. CONTRACT QUALITY ASSURANCE

a. ORIGIN
☐ CQA ☐ ACCEPTANCE of listed items has been made by me or under my supervision and they conform to contract, except as noted herein or on supporting documents.

DATE SIGNATURE OF AUTHORIZED GOVERNMENT REPRESENTATIVE
TYPED NAME:
TITLE:
MAILING ADDRESS:
COMMERCIAL TELEPHONE NUMBER:

b. DESTINATION
[✓] CQA [X] ACCEPTANCE of listed items has been made by me or under my supervision and they conform to contract, except as noted herein or on supporting documents.

DATE SIGNATURE OF AUTHORIZED GOVERNMENT REPRESENTATIVE
TYPED NAME:
TITLE:
MAILING ADDRESS:
COMMERCIAL TELEPHONE NUMBER:

22. RECEIVER'S USE
Quantities shown in column 17 were received in apparent good condition except as noted.

DATE RECEIVED SIGNATURE OF AUTHORIZED GOVERNMENT REPRESENTATIVE
TYPED NAME:
TITLE:
MAILING ADDRESS:
COMMERCIAL TELEPHONE NUMBER:

* *If quantity received by the Government is the same as quantity shipped, indicate by (X) mark; if different, enter actual quantity received below quantity shipped and encircle.*

23. CONTRACTOR USE ONLY
ACRN: AA 97 2040013012RP 6X20P2S10K4255Y S01021 W32G3H

[illegible]

DD FORM 250, AUG 2000 PREVIOUS EDITION IS OBSOLETE.

BEST AVAILABLE COPY